

Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных технологий

Кафедра программного обеспечения вычислительной техники и  
автоматизированных систем

**КУРСОВАЯ РАБОТА**

по дисциплине «Основы искусственного интеллекта»

**Разработка компонентов программно-информационных систем с элемен-  
тами искусственного интеллекта**

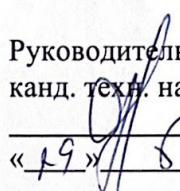
Пояснительная записка

ОГУ 09.03.04.3023.187 ПЗ



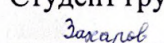
Руководитель

канд. техн. наук, доцент

 А.М. Семенов

« 19 » 2023 г.

Студент группы 20ПИИж(б)РПиС

 А.В. Захаров

« 29 » мая 2023 г.

Оренбург 2023



Утверждаю  
заведующий кафедрой ПОВТАС  
(наименование кафедры)  
Д.В.Горбачев  
подпись инициалы фамилия  
«    »      20   г.

## ЗАДАНИЕ

на выполнение курсовой работы

студенту Засаров Андрей Васильевич  
фамилия, имя, отчество

по направлению подготовки 09.03.04 Программная инженерия  
код, наименование

по Основы искусственного интеллекта  
наименование дисциплины

1 Тема работы «Интеллектуальная программа-помощник выбора передачи на велосипеде»

2 Срок сдачи студентом работы «29» мая 2023 г.

3 Цель и задачи работы.

Цель: Автоматизация процесса принятия решения выбора эффективной передачи на основе нейронной сети.

Задачи: Изучение теоретических сведений; анализ предметной области; анализ существующих аналогов программных средств; выбор моделей данных, метода и инструментальных средств разработки программной системы; проектирование и разработка приложения.

4 Исходные данные к работе. Набор переменных для реализации программы-помощника выбора передачи на велосипеде.

5 Перечень вопросов, подлежащих разработке. Разработать набор переменных объекта в выбранной предметной области; архитектуру, функциональную схему и алгоритм приложения; руководство по установке, администрированию приложения и пользования ПС. Обоснование выбора средств разработки программной системы.

6 Перечень графического материала. Архитектура, функциональная схема, схема алгоритма программного средства и листинг программы.

Дата выдачи и получения задания

Руководитель «15» 02 20   г.

Студент «13» февраля 2023 г.

А.М.Семенов  
подпись инициалы, фамилия  
Засаров  
подпись инициалы, фамилия

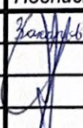


## Аннотация

В данной курсовой работе проведены исследования возможностей по разработке программы-помощника выбора передачи на велосипеде на основе инструмента TensorFlow и мобильного приложения, написанного на языке программирования высокого уровня Kotlin.

В TensorFlow имеется большой набор интегрированных служб, позволяющих создавать или использовать готовые нейронные сети в разных сферах жизни, например: распознавание объектов, распознавание текста, обработка аудио, понимание смысла сообщения для быстрого ответа и т. д. и т. п..

Работа содержит 22 листа текста, 7 рисунков, 0 таблиц, 6 приложений.

ОГУ 09.03.04.3023.187 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата
Разраб.		Захаров А.В.		29.05.22
Провер.		Семенов А.М.		
Н.контр.				
Зав. Каф.		Горбачев Д.В.		

Разработка компонентов программно-информационных систем с элементами искусственного интеллекта		
Лит.	Лист	Листов
К	Р	3
ИМИТ, 20ПИИж(б)РПИС		

## Содержание

Введение .....	5
1 Краткие теоретические сведения .....	6
1.1 Многослойный персептрон .....	6
2 Анализ аналогов.....	8
2.1 Обоснование выбора программных средств разработки. ....	8
3 Руководство по эксплуатации программного средства .....	10
3.1 Требования к системным ресурсам .....	10
3.2 Руководство пользователя.....	10
Заключение.....	12
Список использованных источников.....	13
Приложение А (обязательное) Архитектура программного средства .....	14
Приложение Б (обязательное) Укрупнённая схема алгоритма .....	15
Приложение В (обязательное) Функциональная схема программного средства...	16
Приложение Г (обязательное) Фрагмент обучающей выборки.....	17
Приложение Д (обязательное) Архитектура нейронной сети.....	18
Приложение Е (обязательное) Фрагмент программы .....	19

## Введение

Велосипедный компьютер для оптимальной передачи на основе нейронной сети — это инновационное решение в области велосипедного спорта, которое помогает улучшить эффективность занятий и соревнований. Одним из главных преимуществ такой системы является возможность определения наилучшей передачи в зависимости от условий трассы и физического состояния велосипедиста, что в свою очередь увеличивает скорость и снижает затраты энергии.

Цель данной работы заключается в разработке велосипедного компьютера, основанного на нейронной сети, которая будет способна собирать информацию о скорости, направлении ветра и других параметрах движения в режиме реального времени, и рассчитывать наилучшую передачу для достижения максимальной эффективности.

Среди основных задач работы следует выделить: изучение существующих решений в данной области, создание модели нейронной сети, обработка входных данных, оптимизация процесса подбора наилучшей передачи, а также проведение экспериментов и оценка эффективности разработанной системы.

Разработка велосипедного компьютера для предложения оптимальной передачи на основе нейронной сети имеет явную практическую значимость и востребованность в велоспорте, что делает данную работу актуальной и значимой в настоящее время.

# 1 Краткие теоретические сведения

## 1.1 Многослойный персептрон

Алгоритм обратного распространения ошибки. Самым важным свойством нейронных сетей является их способность обучаться (*learn*) на основе данных окружающей среды и в результате обучения повышать свою производительность. Повышение производительности происходит со временем в соответствии с определенными правилами. *Обучение нейронной сети происходит посредством интерактивного процесса корректировки синаптических весов и порогов.* Различают алгоритмы обучения с учителем и без учителя.

Обучение с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются обучающей парой. Обычно сеть обучается на некотором числе таких обучающих пар. Предъявляется входной вектор, вычисляется выход сети и сравнивается с соответствующим целевым вектором, разность (ошибка) с помощью обратной связи подается в сеть, и веса изменяются в соответствии с алгоритмом, стремящимся минимизировать ошибку. *Сигнал ошибки (error signal)* — это разность между желаемым сигналом и текущим откликом нейронной сети. Корректировка параметров выполняется пошагово с целью имитации (*emulation*) нейронной сетью поведения учителя. Концептуально участие учителя можно рассматривать как наличие знаний об окружающей среде, представленных в виде пар вход-выход. При этом сама среда неизвестна обучаемой нейронной сети.

Многослойные персептроны успешно применяются для решения разнообразных сложных задач. При этом обучение с учителем выполняется с помощью такого популярного алгоритма, как алгоритм обратного распространения ошибки (*error back-propagation algorithm*). Этот алгоритм основывается на коррекции ошибок (*error-correction learning rule*).

Процесс обучения НС представлен на рисунке 1.1.

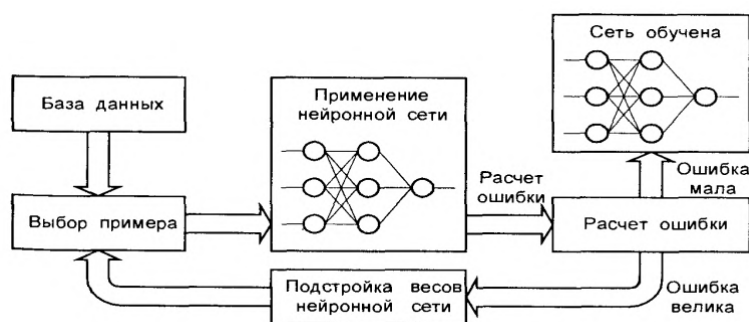


Рисунок 1.1 - Процесс обучения НС

Не существует универсального алгоритма обучения, подходящего для всех архитектур нейронных сетей. Существует лишь набор средств, представленный множеством алгоритмов обучения, каждый из которых имеет свои достоинства. Алгоритмы обучения отличаются друг от друга способом настройки

синаптических весов нейронов. Еще одной отличительной характеристикой является способ связи обучаемой нейросети с внешним миром. В этом контексте говорят о парадигме обучения (*learning paradigm*), связанной с моделью окружающей среды, в которой функционирует данная нейронная сеть.

Пять основных моделей обучения:

- обучение, основанное на коррекции ошибок;
- конкурентном обучении;
- с использованием памяти;
- Хеббовском обучении;
- методе Больцмана.

Обучение, основанное на коррекции ошибок, реализует метод оптимальной фильтрации. Обучение на основе памяти предполагает явное использование обучающих данных. Метод Хебба и конкурентный подход к обучению основаны на нейробиологических принципах. В основу метода Больцмана положены идеи статистической механики.

Обучение методом обратного распространения ошибки предполагает два прохода по всем слоям сети: **прямого и обратного**.

*При прямом проходе (forward pass)* образ (входной вектор) подается на сенсорные узлы сети, после чего распространяется по сети от слоя к слою. В результате генерируется набор выходных сигналов, который и является фактической реакцией сети на данный входной образ. Во время прямого прохода все синаптические веса сети фиксированы.

*Во время обратного прохода (back-ward pass)* все синаптические веса настраиваются в соответствии с правилом коррекции ошибок, а именно: фактический выход сети вычитается из желаемого (целевого) отклика, в результате чего формируется сигнал ошибки (*error signal*). Этот сигнал впоследствии распространяется по сети в направлении, обратном направлению синаптических связей. Отсюда и название - *алгоритм обратного распространения ошибки*.

Синаптические веса настраиваются с целью максимального приближения выходного сигнала сети к желаемому в статистическом смысле.

**Обучение сети обратного распространения требует выполнения следующих операций:**

1. Выбрать очередную обучающую пару из обучающего множества; подать входной вектор на вход сети.
2. Вычислить выход сети.
3. Вычислить разность между выходом сети и требуемым выходом (целевым вектором обучающей пары).
4. Определение ошибки выходного слоя.
5. Определение ошибки скрытого слоя.
6. Коррекция весов.
7. Определение обновлённого веса синапса.
8. Повторить шаги с 1 по 3 (включительно) для каждого вектора обучающего множества до тех пор, пока ошибка на всем множестве не достигнет приемлемого уровня.

## 2 Анализ аналогов

Мобильное приложение Bike Computer - Cycling Tool от разработчика fitzeeee.com представляет собой велокомпьютер, основанный на GPS.

Преимущества:

- Наличие спидометра (из данных, полученных от GPS);
- Подсчёт калорий;
- Пройденное расстояние.

Недостатки:

- Отсутствие контроля текущей передачи;
- Отсутствие учёта скорости и направления ветра;
- Отсутствие рекомендаций о выборе текущей передачи.

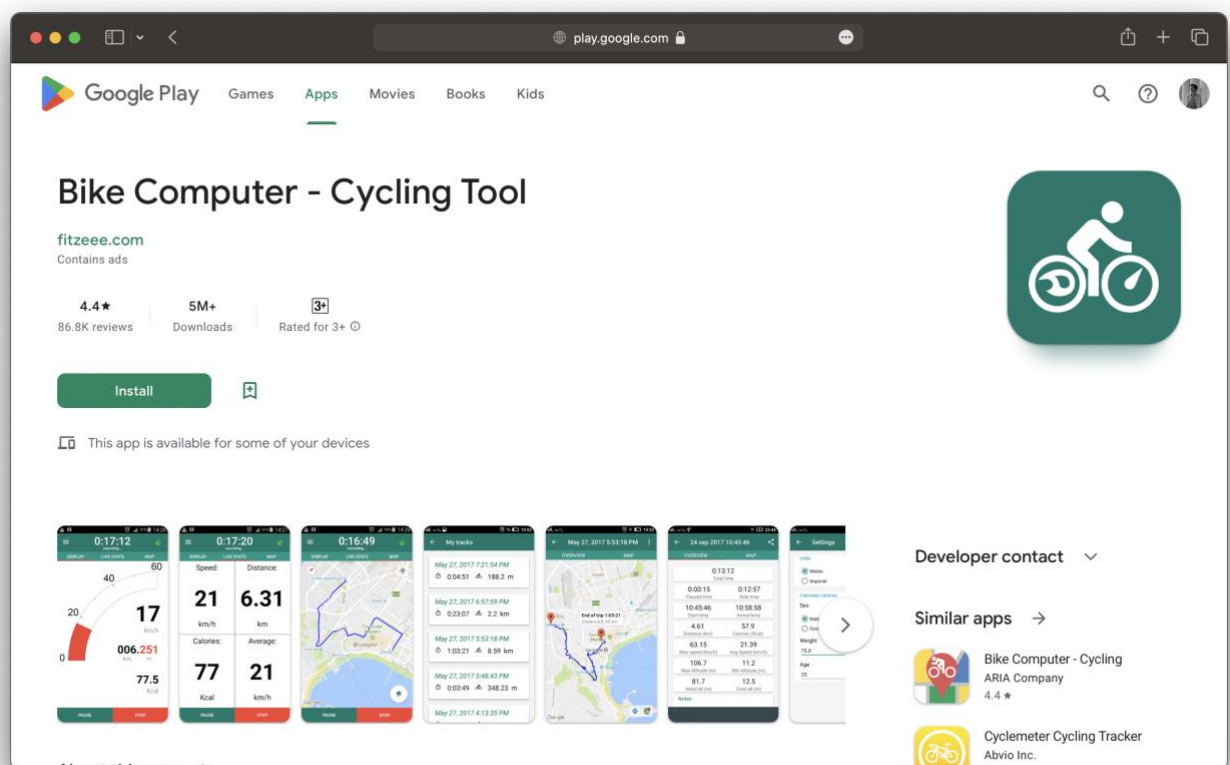


Рисунок 2.1 – Страница приложения в Google Play

### 2.1 Обоснование выбора программных средств разработки.

Что такое Android Studio?

Android Studio — это интегрированная среда разработки приложений для операционной системы Android, созданная на основе IntelliJ IDEA и использующая язык Java. Эта среда разработки поддерживает создание и отладку приложений для мобильных устройств и планшетов на Android.

Какие плюсы у Android Studio?

1. Простота использования, гибкость и расширяемость.
2. Интуитивно понятный интерфейс и хорошая документация.



3. Поддержка большого количества языков программирования, в том числе Kotlin, Java, C++ и других.

4. Встроенный Emulator Android для отладки приложений без реального устройства.

5. Расширенные возможности по профилированию и оптимизации производительности приложений.

Какой язык программирования выбрать для создания велосипедного компьютера?

Kotlin — это статически типизированный язык программирования, разработанный компанией JetBrains, и официально поддерживаемый Google для разработки приложений под Android. Представляет собой развитие языка Java, обладает более высокой читаемостью кода, легче поддаётся компиляции и работает быстрее, чем Java.

Какие плюсы у Kotlin?

1. Повышенная читаемость кода и увеличение производительности разработки благодаря удобному и компактному синтаксису.

2. Интероперабельность с Java, что позволяет использовать API и библиотеки Java.

3. Надёжность и безопасность, так как Kotlin предотвращает некоторые ошибки времени выполнения, которые могут возникнуть при использовании Java.

4. Совместимость с платформой Android и поддержка последних оптимизаций и фреймворков.

Какую библиотеку машинного обучения использовать для оптимизации передач?

TensorFlow Lite — это лёгкий вариант фреймворка TensorFlow, который предназначен для использования на мобильных устройствах. TensorFlow Lite позволяет запустить обученную модель машинного обучения в приложении на Android без необходимости подключения к Интернету.

Какие плюсы у TensorFlow Lite?

1. Высокая производительность и эффективность работы на мобильных устройствах, благодаря оптимизациям и минимальным требованиям к ресурсам.

2. Поддержка широкого спектра задач машинного обучения, в том числе классификация изображений, сегментация, детектирование объектов и многое другое.

3. Лёгкость интеграции с приложением на Android и хороший уровень документации.

4. Возможность запуска на устройстве без подключения к интернету, что обеспечивает более высокую конфиденциальность и защиту персональных данных пользователей.

### 3 Руководство по эксплуатации программного средства

#### 3.1 Требования к системным ресурсам

Для работы ПП требуется смартфон на базе операционной системы Android, удовлетворяющий следующей минимальной конфигурации:

- Версия Android не ниже 6.0;
- Наличие сервисов Google;
- Наличие GPS;
- Наличие датчика компаса.

Программа занимает 80 Мб постоянной памяти.

#### 3.2 Руководство пользователя

Для запуска программы необходимо нажать на иконку программы. При первом запуске приложения появится системный запрос на использование геопозиции (см. рисунок 3.1). Данное разрешение позволяет определять скорость.

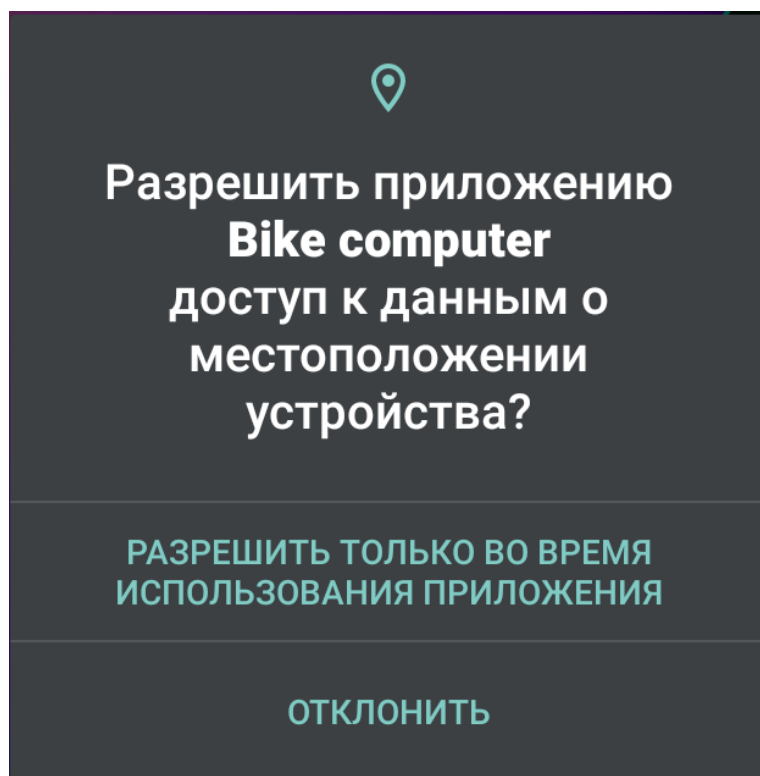


Рисунок 3.1 – Системный запрос на доступ к данным о местоположении устройства

Всё приложение состоит из главного окна (см. рис. 3.2). Число посередине – текущая скорость (в км/ч), элементы по бокам – блоки отвечающие за левый переключатель передачи и правый соответственно. Каждый блок состоит из кнопок повышения / понижения передачи, большого числа – текущая передача, маленького числа – рекомендуемая передача.

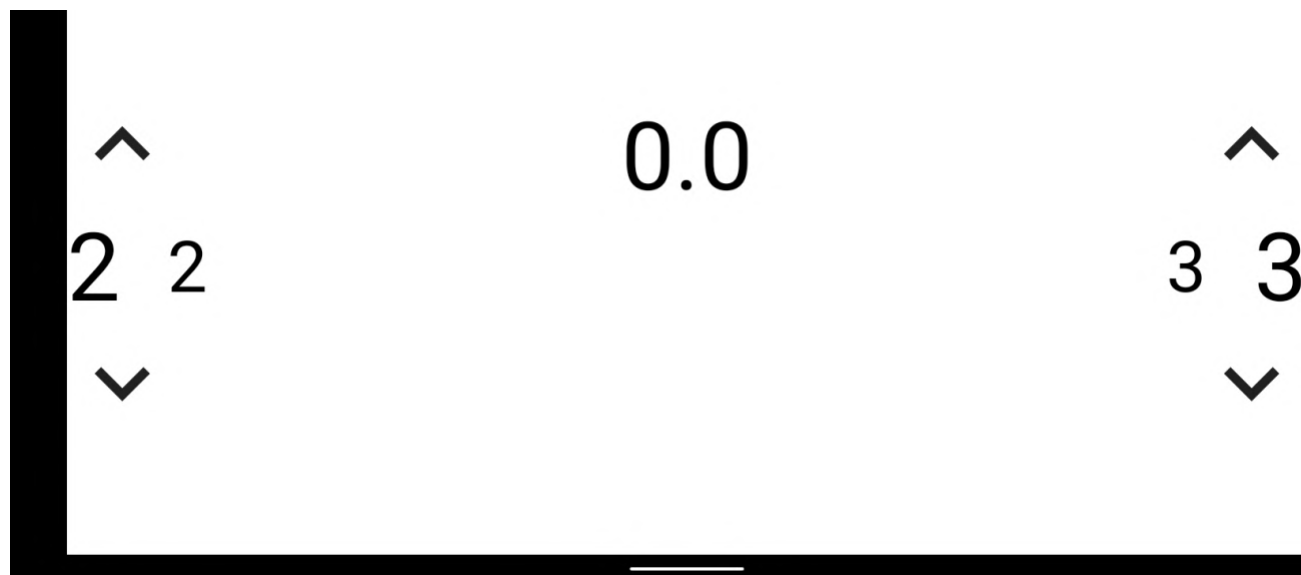


Рисунок 3.2 – Окно приложения

## Заключение

Рассмотрим результаты выполнения задания, целью которого являлась разработка велосипедного компьютера, основанного на нейронной сети, которая будет способна собирать информацию о скорости, направлении ветра и других параметрах движения в режиме реального времени, и рассчитывать наилучшую передачу для достижения максимальной эффективности.

Выполнены следующие задачи:

- изучили существующие решения в данной области;
- создали модели нейронной сети;
- обработали входные данные;
- оптимизировали процесс подбора наилучшей передачи;
- провели эксперименты;
- оценили эффективность разработанной системы.

Таким образом, поставленные инженерные задачи выполнены, цель курсовой работы достигнута.



## Список использованных источников

1 Семенов, А.М. Интеллектуальные системы [Текст] : учебное пособие для студентов, обучающихся по программам высшего профессионального образования по направлениям подготовки 230100.68 Информатика и вычислительная техника, 231000.68 Программная инженерия / А. М. Семенов, Н. А. Соловьев, Е. Н. Чернопрудова, А. С. Цыганков; М-во образования и науки Рос. Федерации, Федер. гос. бюджет. образоват. учреждение высш. проф. образования "Оренбург. гос. ун-т". - Оренбург : ОГИМ, 2014. - 237 с. - Библиогр.: с. 218-221. - Прил.: с. 222-236. - ISBN 978-5-9723-0158-4. Издание на др.носителе [Электронный ресурс]

2 Семенов, А.М. Интеллектуальные системы [Электронный ресурс] : учебное пособие для студентов, обучающихся по программам высшего профессионального образования по направлениям подготовки 230100.68 Информатика и вычислительная техника, 231000.68 Программная инженерия / А. М. Семенов [и др.]; М-во образования и науки Рос. Федерации, Федер. гос. бюджет. образоват. учреждение высш. проф. образования "Оренбург. гос. ун-т". - Электрон. текстовые дан. (1 файл: 3.85 Мб). - Оренбург : ОГУ, 2013. - 236 с. - Загл. с тит. экрана. -Adobe Acrobat Reader 6.0-ISBN978-5-9723-0158-4.. Издание на др. носителе [Текст]

3 Kotlin [Электронный ресурс] // Kotlin. - Режим доступа: <https://kotlinlang.org>.

4 Android Studio [Электронный ресурс] // Android Studio. - Режим доступа: <https://developer.android.com/studio>.

5 TensorFlow Lite [Электронный ресурс] // TensorFlow Lite. - Режим доступа: <https://www.tensorflow.org/lite>.

## Приложение А (обязательное)

### Архитектура программного средства

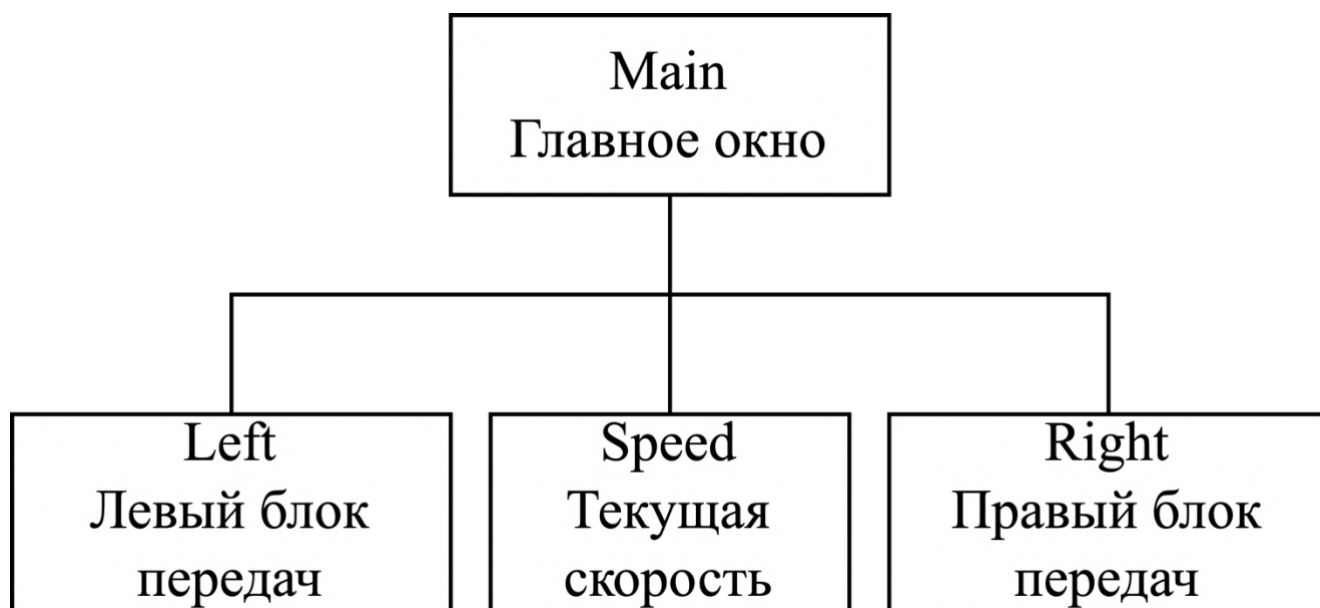


Рисунок А.1 – Архитектура программного средства

## Приложение Б (обязательное)

### Укрупнённая схема алгоритма



Рисунок Б.1 – Укрупнённая схема алгоритма

## Приложение В (обязательное)

### Функциональная схема программного средства

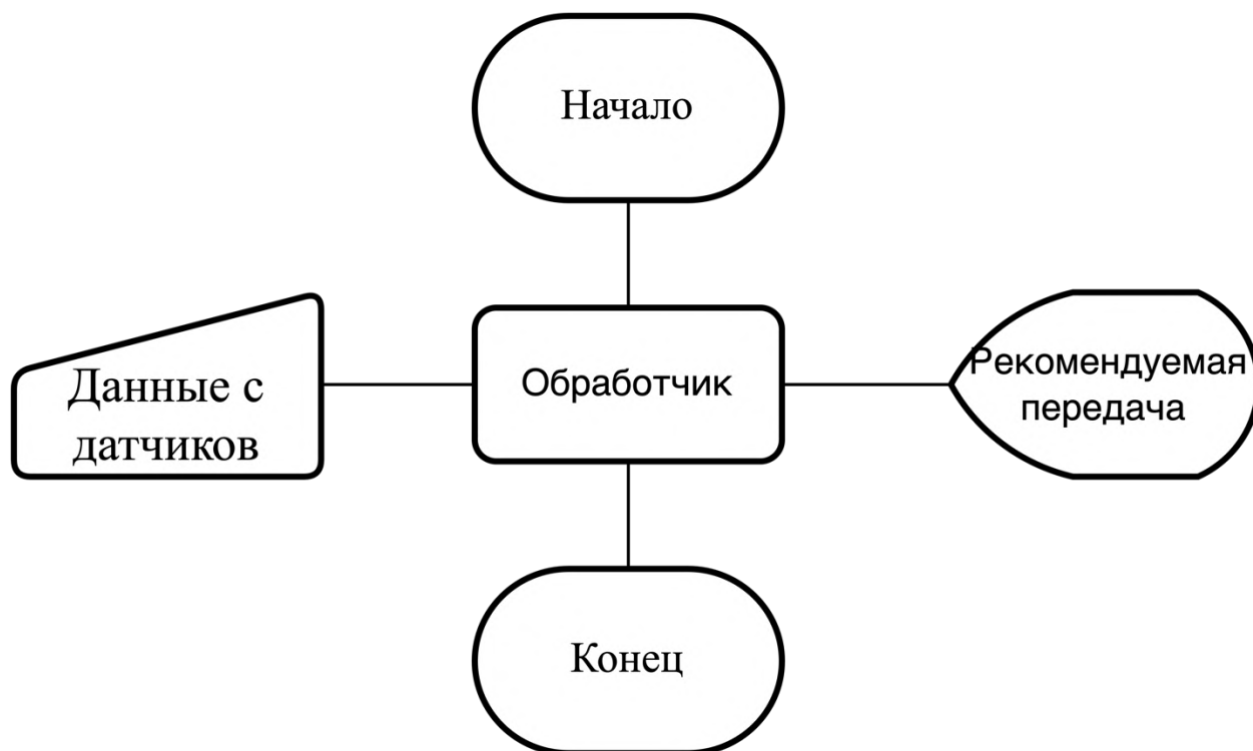


Рисунок В.1 - Функциональная схема программного средства



## Приложение Г (обязательное)

### Фрагмент обучающей выборки

0 26.41 0.289271 167 71 17.3 8  
0 26.27 -0.134097 177 81 17.3 8  
0 26.27 -0.440302 197 101 17.3 8  
0 27.0 0.411284 177 81 17.3 8  
0 27.0 0.029836 147 51 17.3 8  
0 26.91 -0.366951 205 109 17.3 8  
0 26.91 0.489918 215 119 17.3 8  
0 26.57 -0.01063 165 69 17.3 8  
0 26.57 0.241903 161 65 17.3 8  
0 27.09 -0.556054 184 88 17.3 8  
0 27.09 -0.982046 190 94 17.3 8  
0 27.58 0.163661 174 78 17.3 8  
0 27.58 0.325617 156 60 17.3 8  
0 27.46 -0.4672 169 73 17.3 8  
0 27.46 -0.141246 185 89 17.3 8  
0 27.52 -0.089052 176 80 17.3 8  
0 27.52 -0.048951 179 83 17.3 8  
0 27.83 -0.194914 180 84 17.3 8  
0 27.83 -0.258435 202 106 17.3 8  
0 27.87 -0.397092 168 72 17.3 8  
0 27.87 0.59294 146 50 17.3 8  
0 27.92 0.304674 158 62 17.3 8  
0 27.92 -0.214664 160 64 17.3 8  
0 27.02 -0.108634 136 40 17.3 8  
0 27.02 0.120557 160 64 17.3 8  
0 27.52 -0.136058 187 91 17.3 8  
0 27.52 0.454883 176 80 17.3 6  
0 26.91 -0.00191 166 70 17.3 5  
0 26.91 -0.012914 167 71 17.3 5  
1 26.03 -0.26309 160 64 17.3 5  
1 26.03 0.004451 149 53 17.3 5  
1 24.52 -0.095874 180 84 17.3 5  
1 24.52 -0.372805 152 56 17.3 5  
1 22.99 0.144943 151 55 17.3 5  
1 22.99 -0.076247 164 68 17.3 5  
1 21.33 0.068317 177 81 17.3 5  
1 21.33 0.214724 146 50 17.3 5  
1 19.95 0.795255 234 138 17.3 5  
1 19.95 0.386768 142 46 17.3 5

## **Приложение Д** *(обязательное)*

### **Архитектура нейронной сети**

Один скрытый слой с 64 нейронами – активация relu  
Выходной слой с 10 нейронами – активация softmax

## Приложение Е (обязательное)

### Фрагмент программы

```
package com.example.bikecomputer

import android.annotation.SuppressLint
import android.content.Context
import android.content.Context.MODE_PRIVATE
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import android.location.Location
import android.location.LocationListener
import android.location.LocationManager
import android.util.Log
import com.example.bikecomputer.weather.HourRepository
import com.example.bikecomputer.weather.models.Hour
import io.ktor.util.date.*
import kotlinx.coroutines.*
import kotlinx.coroutines.channels.ProducerScope
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.channelFlow
import org.tensorflow.lite.Interpreter
import org.tensorflow.lite.support.common.FileUtil
import java.io.FileOutputStream
import kotlin.math.PI
import kotlin.math.abs
import kotlin.math.roundToInt

@SuppressLint("MissingPermission")
class MainWorker(
    private val appContext: Context,
    private val hourRepository: HourRepository
) : LocationListener, SensorEventListener {

    private val leftShiftValues = 1..3
    private val rightShiftValues = 1..8

    private val locationManager: LocationManager =
        appContext.getSystemService(Context.LOCATION_SERVICE) as LocationManager
    private val sensorManager: SensorManager =
        appContext.getSystemService(Context.SENSOR_SERVICE) as SensorManager

    private var brake: Boolean = false
    private var leftShift: Byte = 2
    private var rightShift: Byte = 3

    private var speed: Float = 0.0f
    private var gyroscope: Float = 0f

    private var accelerometer: FloatArray? = null
    private var magneticField: FloatArray? = null

    private var hours: Array<Hour>? = null
    private var windAngleNow: Int? = null
    private var windSpeedNow: Double? = null

    private lateinit var workerState: WorkerState
```

```

private lateinit var fileStream: FileOutputStream

fun start(): Flow<WorkerState> {
    registerListeners()
    val nameFile = "${getTimeMillis()}.txt"
    fileStream = applicationContext.openFileOutput(nameFile, MODE_PRIVATE)
    return channelFlow {
        launch {
            withContext(Dispatchers.Default) {
                mainWork()
            }
        }
    }
}

private fun registerListeners() {
    locationManager.requestLocationUpdates(
        locationManager.GPS_PROVIDER,
        500,
        0f,
        this
    )

    val sensorsList = listOf(
        Sensor.TYPE_GYROSCOPE,
        Sensor.TYPE_ACCELEROMETER,
        Sensor.TYPE_MAGNETIC_FIELD
    )

    sensorsList.forEach {
        sensorManager.registerListener(
            this,
            sensorManager.getDefaultSensor(it),
            SensorManager.SENSOR_DELAY_GAME
        )
    }
}

private suspend fun ProducerScope<WorkerState>.mainWork() {
    /*ourRepository.update().onEach {
        if (it is Resource.Success) {
            hours = it.data
            updateWindData()
        }
    }.launchIn(this)*/
    hours = hourRepository.get().data
    updateWindData()
    while (true) {
        updateWorkerState()
        //writeToFile()
        runTensorflow()
        send(workerState)
        delay(500)
    }
}

private fun updateWorkerState() {
    val compass = getCompass()
    workerState = WorkerState(
        brake,
        leftShift,
        rightShift,

```



```

        null,
        null,
        getSpeed(),
        gyroscope,
        compass,
        getAngle(compass),
        windSpeedNow
    )
}

private fun runTensorflow() {
    if (workerState.angle == null || workerState.windSpeed == null) return
    try {
        val interpreter = Interpreter(FileUtil.loadMappedFile(appContext,
"model.tflite"))
        val input = floatArrayOf(
            if (workerState.brake) 1.0f else 0.0f,
            workerState.speed.toFloat(),
            workerState.gyroscope,
            workerState.angle!!.toFloat(),
            workerState.windSpeed!!.toFloat()
        )
        val output = Array(1) {
            FloatArray(9) { .0f }
        }
        interpreter.run(input, output)
        val max = output[0].max()
        val shift = output[0].indexOfFirst { it == max } + 2
        val res = when (shift) {
            in 0..3 -> Pair(1, shift)
            in 4..7 -> Pair(2, shift - 1)
            else -> Pair(3, shift - 2)
        }
        workerState.recommendedLeftShift = res.first.toByte()
        workerState.recommendedRightShift = res.second.toByte()
    } catch (e: Exception) {
        Log.d("ttt", e.message?: "Error")
    }
}

private fun getAngle(compass: Int?): Int? {
    if (compass == null || windAngleNow == null) return null
    var res = abs(compass - windAngleNow!!)
    if (res > 180) {
        res = 360 - res
    }
    return res
}

private fun updateWindData() {
    hours?.let {
        val timeNow = getTimeMillis() / 1000
        for (i in it.indices) {
            if (it[i + 1].timeEpoch > timeNow) {
                windAngleNow = it[i].windDegree
                windSpeedNow = it[i].windKph
                break
            }
        }
    }
}

/*private fun writeToFile() {

```

```

        var text = ""
        workerState.apply {
            text += if (brake) 1 else 0
            text += " $speed"
            text += " $gyroscope"
            text += " $compass"
            text += " $angle"
            text += " $windSpeed"
            text += " $shift\n"
        }
        fileStream.write(text.toByteArray())
    }*/

    override fun onLocationChanged(location: Location) {
        speed = location.speed
    }

    override fun onSensorChanged(event: SensorEvent?) {
        if (event == null) return
        when (event.sensor.type) {
            Sensor.TYPE_GYROSCOPE -> {
                gyroscope = event.values[0]
            }
            Sensor.TYPE_ACCELEROMETER -> {
                accelerometer = event.values
            }
            Sensor.TYPE_MAGNETIC_FIELD -> {
                magneticField = event.values
            }
        }
    }

    private fun getSpeed(): Double = (speed * 3.6f * 100).roundToInt() / 100.0

    private fun getCompass(): Int? {
        if (accelerometer != null && magneticField != null) {
            val R = FloatArray(9)
            val I = FloatArray(9)
            val success = SensorManager.getRotationMatrix(R, I, accelerometer,
magneticField)
            if (success) {
                val orientation = FloatArray(3)
                SensorManager.getOrientation(R, orientation)
                val azimuth = orientation[0]
                return ((azimuth / PI * 180).toInt() + 450) % 360
            }
        }
        return null
    }

    fun brakeChanged(isDown: Boolean) {
        brake = isDown
    }

    fun leftShiftChanged(isUp: Boolean): Byte {
        val newValue: Byte = (leftShift + if (isUp) 1 else -1).toByte()
        if (newValue in leftShiftValues) leftShift = newValue
        return leftShift
    }

    fun rightShiftChanged(isUp: Boolean): Byte {
        val newValue: Byte = (rightShift + if (isUp) 1 else -1).toByte()
        if (newValue in rightShiftValues) rightShift = newValue
    }

```

```
        return rightShift
    }

    private fun getShift(): Byte = (rightShift + leftShift - 1).toByte()

    fun stop() {
        fileStream.close()
        locationManager.removeUpdates(this)
        sensorManager.unregisterListener(this)
    }

    override fun onAccuracyChanged(sensor: Sensor?, accuracy: Int) {}
}
```